|  |  |
| --- | --- |
| TEA  PRIMITIVE | SEMANTICS |
| C: | |  |  | | --- | --- | | NAME | Clear | | PURPOSE | Clear working memory | | SYNTAX  & SEMANTICS | |  | | --- | | c: | | Set Empty String as IO/AI | | c:PARAMS | | INERT | | c!: | | Set Empty String as IO and set to Empty String all  currently active vaults (including the DEFAULT VAULT) | | c!:PARAMS | | INERT | | c\*: | | INERT | | c\*:v1:v2:…:vN  c\*!:v1:v2:…:vN | | Clear the vaults specified v1, v2,…, vN ONLY. Does NOT tamper with IO/AI | |  | | NOTES | The **Empty String** is the sequence “” – a string of 0 length.  By the definitions above, the program:  i!:{BC}| c:  # (=, VAULTS:{})  Should return “”. While  i!:{BC} | v: | c: | y:  # (=BC, VAULTS:{"":"BC"})  Should return “BC”. But  i!:{BC} | v: | v:XX:{T} | c!: | y:XX  # (=, VAULTS:{"":"","XX":""})  Shall also return “” because whether or not we read the main memory (AI) or any variable memory (such as the default vault “” or the “XX” vault written to in this example program), after a call to **c!:**, all active memory is essentially cleared.  For cases where one wishes to clear only specific sections of memory, via named vaults, the following example can be illustrative enough:  i!:{BC} | v: | c: | y: | a!: | v:vA | n:100 | v:vB | v:vGLUE:\*\* | c\*!:vA | g\*!:vGLUE:vA:vB  Should return "\*\*N" where N is some random number. Otherwise, without c\*!:vA, such as in the following version:  i!:{BC} | v: | c: | y: | a!: | v:vA | n:100 | v:vB | v:vGLUE:\*\* | g\*!:vGLUE:vA:vB  should have return something like CB\*\*N or BC\*\*N |  | |